Name : W. K. D. D. Senuruk
Index Number : 190586H

EN2550 Homework 1 on Python and NumPy

(1)

In [1]:
```
for i in range(1,6):
    print(i, ' : ', (i)**(2))
```

```
1 : 1
2 : 4
3 : 9
4 : 16
5 : 25
```

(2)

In [2]:
```
import sympy

for i in range(1,6):
    if not sympy.isprime(i):
        print(i, ' : ', (i)**(2))
```

```
1 : 1
4 : 16
```

(3)

In [3]:
```
ans = [i**(2) for i in range(1, 6)]
print(ans)
```

```
[1, 4, 9, 16, 25]
```

(4)

In [4]:
```
ans = [i**(2) for i in range(1,6) if not sympy.isprime(i)]
print(ans)
```

```
[1, 16]
```

(5) (a)

In [5]:
```
import numpy as np

A = np.array([[1, 2], [3, 4], [5, 6]])
B = np.array([[7, 8, 9, 1], [1, 2, 3, 4]])
C = np.matmul(A, B)
print(C)

[[ 9 12 15  9]
 [25 32 39 19]
 [41 52 63 29]]
```

(5) (b)

In [6]:
```
A = np.array([[1, 2], [3, 4], [5, 6]])
B = np.array([[3, 2], [5, 4], [3, 1]])
D = np.multiply(A, B)
print(D)

[[ 3  4]
 [15 16]
 [15  6]]
```

(6)

In [7]:
```
E = np.random.rand(5, 7)
F = 10 * E
for i in range(5):
    for j in range(7):
        F[i][j] = round(F[i][j])
print(F,'\n')

G = F[1:4, :2]
print(G, '\n')

print("Size of the Resulting Array : 3 x 2")
```

```
[[ 0.  1.  4.  1.  1.  9.  4.]
 [ 3.  2.  9.  0.  0.  5.  6.]
 [ 5.  6.  4.  6.  8.  1.  7.]
 [ 6.  2.  4.  8. 10.  9.  1.]
 [ 7.  2.  0.  0.  6.  8.  1.]]

[[3. 2.]
 [5. 6.]
 [6. 2.]]

Size of the Resulting Array : 3 x 2
```

(7)

In [8]:
```
x = np.array([[1, 2, 3], [4, 5, 6]])
y = np.array([10, 100])
z = np.array([10, 100, 1000])
k = 10

print('_____Broadcasting_____')
print('multiply a matrix by a constant')
print(x * k)

print('Add a vector to each row')
print(x + z)

print('Add a vector to each column')
print(x + np.reshape(y, (2, 1)))
```

```
_____Broadcasting_____
multiply a matrix by a constant
[[10 20 30]
 [40 50 60]]
Add a vector to each row
[[  11  102 1003]
 [  14  105 1006]]
Add a vector to each column
[[ 11  12  13]
 [104 105 106]]
```

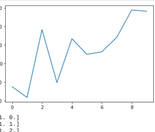
(8) (a)

In [9]:
```
import matplotlib.pyplot as plt

m, c = 2, -4
N = 10
x = np.linspace(0, N - 1, N).reshape(N, 1)
sigma = 10
y = m*x + c + np.random.normal(0, sigma, (N, 1))

plt.plot(x, y)
plt.show()

X = np.append(np.ones((N, 1)), x, axis = 1)
print(X)
```



```
[[1. 0.]
 [1. 1.]
 [1. 2.]
 [1. 3.]
 [1. 4.]
 [1. 5.]
 [1. 6.]
 [1. 7.]
 [1. 8.]
 [1. 9.]]
```

(8) (b)

In [10]:
```
ans = np.linalg.inv(X.T @ X) @ X.T @ y
print(ans)
```

```
[[-12.26964242]
 [  4.3520699 ]]
```

(9) (a)

In [11]:
```
def sqrt_hyper_est(s):
    if s > 100:
        a = "{:e}".format(s)
        idx = a.index('+')
        n = int(float(a[idx + 1:]))
        if n % 2 == 0:
            a = float(a[:idx - 1])
        else:
            n = n - 1
            a = float(a[:idx - 1]) * 10
        sqrtS = (((-190)/(a + 20)) + 10) * 10**(n//2)
    else:
        sqrtS = (((-190)/(s + 20)) + 10) * 10**(0)

    return sqrtS
```

(9) (b)

In [12]:
```
def sqrt_new_rap(s):
    x = sqrt_hyper_est(s)
    tolerance = 1e-5
    epsilon = 1e-14
    maxIterations = 20
    solutionFound = False

    for i in range(20):
        y = x**(2) - s
        yp = 2*x
        if abs(yp) < epsilon:
            break
        xx = x - y/yp
        if abs(xx - x) <= tolerance:
            solutionFound = True
            break
        x = xx

    if solutionFound == True:
        print('Square root of', s, 'is', xx)
    else:
        print('Not converging')
```

(9) (c)

In [13]:
```
print('_____Square roots for a precision of 0.00001_____')
sqrt_new_rap(64)
sqrt_new_rap(75)
sqrt_new_rap(100)
sqrt_new_rap(1600)
```

```
_____Square roots for a precision of 0.00001_____
Square root of 64 is 8.000000000000994
Square root of 75 is 8.660254037844386
Square root of 100 is 10.0
Square root of 1600 is 40.0
```

(10)

In [14]:
```
import cv2 as cv

img = cv.imread(r'./Images/gal_gaussian.png')
gBlur = cv.GaussianBlur(img, (5, 5), 0)

cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', img)
cv.waitKey(5000)
cv.imshow('Image', gBlur)
cv.waitKey(5000)
cv.destroyAllWindows()
```

(11)

In [15]:
```
img = cv.imread(r'./Images/gal_sandp.png')
mBlur = cv.medianBlur(img, 5)

cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', img)
cv.waitKey(5000)
cv.imshow('Image', mBlur)
cv.waitKey(5000)
cv.destroyAllWindows()
```

(12)

In [16]:
```
img = np.zeros((40, 60), dtype = np.uint8)

img[0:21, 30:61] = 125

cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', img)
cv.waitKey(5000)
cv.destroyAllWindows()

fig, ax = plt.subplots()
ax.imshow(img, cmap = 'gray', vmin = 0, vmax = 255)
plt.show()
```

(13)

In [17]:
```
img = np.zeros((40, 60, 3), dtype = np.uint8)

img = cv.cvtColor(img, cv.COLOR_BGR2RGB)

img[20:41,0:31] = (132, 24, 218)
cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', img)
cv.waitKey(5000)
cv.destroyAllWindows()

img[20:41,0:31] = (218, 24, 132)
fig, ax = plt.subplots()
ax.imshow(img, cmap = 'brg')
plt.show()
```


(14)

In [18]:
```
img = cv.imread(r'./Images/tom_dark.jpg')

cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', img)
cv.waitKey(5000)

hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

for x in range(0, len(hsv)):
    for y in range(0, len(hsv[0])):
        hsv[x, y][2] += 120

img = cv.cvtColor(hsv, cv.COLOR_HSV2BGR)

cv.imshow('Image', img)
cv.waitKey(5000)
cv.destroyAllWindows()
```